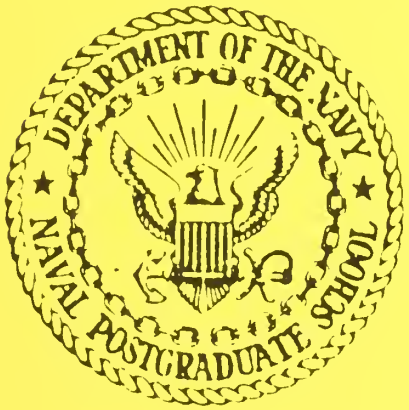
NPS52-86-012

# NAVAL POSTGRADUATE SCHOOL

## Monterey, California




# DATA MODELING ABSTRACTIONS AND GRAPHICAL INTERFACE FOR SUPPORTING THE CONSTRUCTION DESIGN PROCESS

CPT Dana E. Madison

C. Thomas Wu

February 1986



Prepared for: Chief of Naval Research
Arlington, VA 22217

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral R. H. Shumaker
Superintendent

D. A. Schrady
Provost

The work reported herein was supported by Contract from the Office of Naval Research.



This report was prepared by:

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER: NPS52-86-012 | 2. GOVT ACCESSION NO. |
| 3. RECIPIENT'S CATALOG NUMBER | |
| 4. TITLE *(and Subtitle)*: DATA MODELING ABSTRACTIONS AND GRAPHICAL INTERFACE FOR SUPPORTING THE CONSTRUCTION DESIGN PROCESS | 5. TYPE OF REPORT & PERIOD COVERED |
| | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s): CPT Dana E. Madison<br>C. Thomas Wu | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61152N; RR000-01<br>N0001486WR4E001 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Chief of Naval Research<br>Arlington, VA 22217 | 12. REPORT DATE: February 1986 |
| | 13. NUMBER OF PAGES: 20 |
| 14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* | 15. SECURITY CLASS. *(of this report)* |
| | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Database technology has been successfully applied to the traditional data processing environment where data are represented by well-formatted records. There is a growing interest in extending this database technology to more advanced application environments such as VLSI CAD/CAM, cartography, etc., where data are less structured and have very complex semantics. We believe the first step in developing a complete information support system for advanced applications is to identify abstraction concepts which can precisely capture

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

the semantics of an application environment and then to formulate requirements for user interaction with such a system.

In this paper, we identify the necessary abstraction concepts and propose a graphical user interface for an information support system geared towards the house construction design process. An overall goal of our research is to develop a completely integrated information support system for more generic design and manufacturing processes.

# Data Modeling Abstractions and Graphical Interface for Supporting the Construction Design Process

CPT Dana E. Madison

and

C. Thomas Wu

DEPARTMENT OF COMPUTER SCIENCE
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

February 12, 1986

## Abstract

Database technology has been successfully applied to the traditional data processing environment where data are represented by well-formatted records. There is a growing interest in extending this database technology to more advanced application environments such as VLSI CAD/CAM, cartography, etc., where data are less structured and have very complex semantics. We believe the first step in developing a complete information support system for advanced applications is to identify abstraction concepts which can precisely capture the semantics of an application environment and then to formulate requirements for user interaction with such a system.

In this paper, we identify the necessary abstraction concepts and propose a graphical user interface for an information support system geared towards the house construction design process. An overall goal of our research is to develop a completely integrated information support system for more generic design and manufacturing processes.

## Introduction

Traditional hierarchical, relational, and network data models are oriented toward manipulation of logical records and do not fully support the CAD/CAM/CIM design environment. These traditional models lack facilities for handling the semantics which are a component of the design process. There is considerable interest in expanding the use of database technology to support data which is less structured, less formatted, with more complex data types, which would permit modeling of application semantics. Semantic data models attempt to provide high-level data structuring features to improve the expressiveness of database conceptual schemas. This is done by embedding the semantics of a particular application in the database schema. The overall objective of the semantic models is to increase database accessibility by end users, many of whom are not trained in computer science.

In addition to providing for the representation of these semantics, the ideal CAD/CAM/CIM data model would provide other features which are not found in the traditional or current semantic models. One of these features is the representation of design objects as primitives in the model, with prescribed "rules" for associating objects with one another. These objects could be the building blocks from which more complex objects could be built. Operations defined for the data model would include those for manipulating objects. These operations would include provisions for adding new objects and modifying existing ones. Another desirable feature would be a graphics interface to display and manipulate a design. In this paper we will develop a data model which has these desirable features.

The model we propose will be demonstrated by use of a specific example. The example we chose is house design and construction. Much of the application emphasis of CAD/CAM/CIM work done to date has been in the VLSI design process [BAT085], [LEE83], [MCLE83]. We believe that the integrated Information System concept can be extended to more predominant applications, such as automobile and house design. The house design and construction example was chosen because 1) it is an

application that everyone can relate to, and 2) it exemplifies the design and manufacturing process, serving as a generic application.

The major contributions of this paper are the extension to more predominant areas mentioned previously and the inclusion of a graphics user interface in the integrated information system. To the best of our knowledge, this paper is the first attempt to do both of these things. Other research projects have utilized textual query languages [HARD86] or have restricted themselves to specific application areas which are not easily generalized to the generic design environment.

The objective of this paper is to identify the abstraction concepts which are necessary for this design process, and the types of support that a truly integrated system of this kind should provide. The paper is organized as follows: the proposed model will be defined by describing a set of abstraction concepts, comparing them with existing concepts. Next, a graphics-oriented user interface will be discussed. Finally, the paper will conclude with some directions for further research.

## The Model

Current semantic models include the Entity-Relationship (ER) Model [CHEN76], Functional Model, SHM+, SDM/Event Model, TAXIS, and RM/T [BROD82]. All of these models use primitives such as entities, events, or simply objects. They also include provisions for composite objects and attribute specification among the supported features. Extended semantic models integrate a number of programming language concepts with database concepts. They also make use of advanced data type concepts such as abstract data types and strong typing. These extended models include SHM+, TAXIS, and the SDM/Event Model. Semantic modeling theory is now being applied to particular application areas such as office automation, VLSI, and cartography, as well as for traditional data processing applications (inventory, insurance, banking).

The abstraction concepts supported by our model include molecular aggregation, generalization/specialization, version generalization, version hierarchy, instantiation, and instance hierarchy. We believe these abstraction concepts are necessary to support the house construction design process, and therefore are necessary for other advanced application areas as well. Each concept will be described in the remainder of this section.

Molecular aggregation is the abstraction of a set of objects and their relationships into a higher-level object [SMIT77]. This abstraction allows a view of objects from different levels of generality, each with its own level of detailed definition. A user interested in the overall design could use the topmost level of abstraction, which would hide the implementation details. This implements the "Information Hiding" principle commonly found in programming language design. The idea is to give the user only the amount of implementation detail he needs for a particular application. Figure 1 depicts a bathroom as a molecular aggregation of objects called floor, wall, ceiling, sink, toilet, and bathtub. All of these except toilet are molecular objects. Note that several levels of molecular aggregation abstraction are present in the figure.

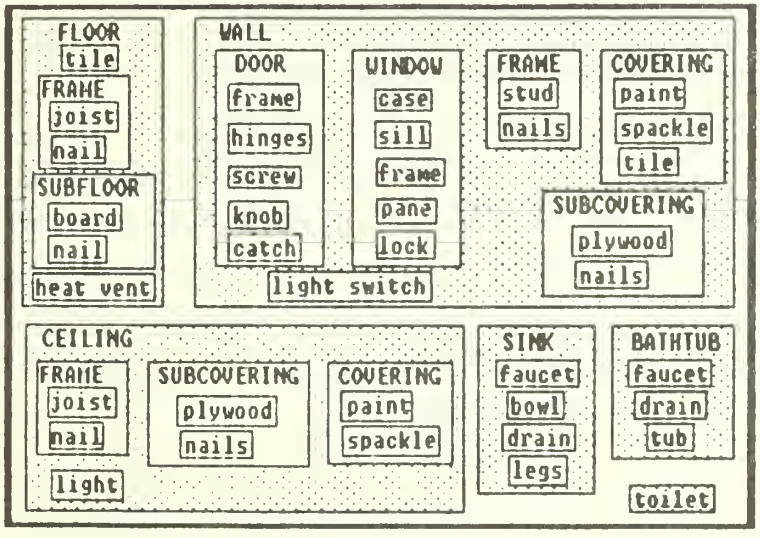



figure 1

The objects whose name appears in upper case are molecular aggregations. Those in lower case represent primitive objects in this example.

Molecular objects have two description components, an interface, and an implementation [BAT085]. The interface specifies the general function of the object and the implementation provides the details of the use of the object for a particular application. Attributes and relationships can be specified in either or both the interface and implementation.

The generalization/specialization concept of Smith and Smith [SMIT77] will be used in the model to provide the relationship between types and their subtypes. Types will be defined either as generalizations of a set of named subtypes, or as primitives from which versions and instances can be made directly. An example of generalization would be the creation of a type house from the subtypes colonial, duplex, ranch, tri-level, and rambling. The notion of subtype is important to the model because different subtypes will be permitted to have different sets of attributes.

Version generalization [BAT085] is used as the mechanism for specifying the relationship each object type has with its versions. A version is created by specifying implementation details for the object type. The concept of parameterized versions [BAT085] arises from the need for allowing freedom in specifying the implementation details for a particular object. If an instance of an object type is defined instead of an instance of one of its versions, a parameterized version is created. Choosing an instance of an object type T creates a socket which will accommodate any version of type T. Using the terminology defined in [BAT085], the different versions are plugged into the socket, creating unique implementations.

The generalization concept of Smith and Smith [SMIT77] differs from version generalization in that the former takes two or more object sets and forms a higher level object set by taking their union, and in the latter, an object type (or subtype) is an abstraction of

the common features of its versions, which is clearly not a union of object sets.

In our model, a version of a type (or subtype) will be defined to be a molecular object with interface details completely specified, but with implementation details in some stage of completion. This definition allows a version to be plugged, partially plugged, or unplugged. Figure 2 shows an object of type A with an object version V1 of type A. The object of type A has its interface defined, which is denoted by the shading of the interface block. The implementation details for this object are not specified, denoted by the unshaded implementation block.

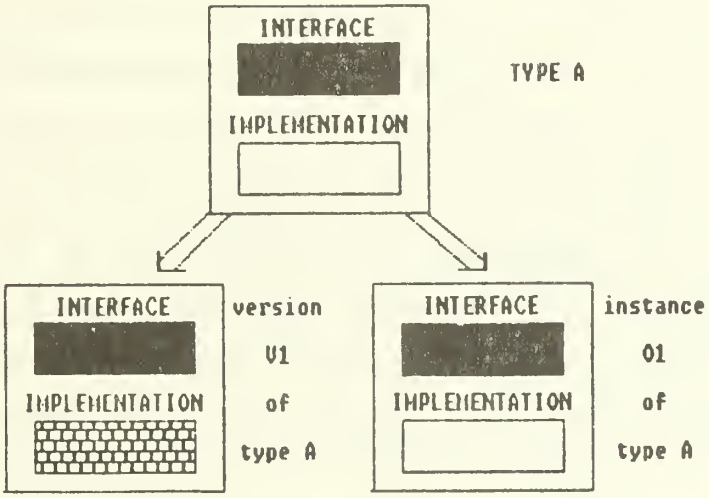



figure 2

Object V1 has the same interface details as its object type, and also has some implementation details specified, denoted by the partially shaded implementation block. Examples of this definition of version are the two, three, and four bedroom versions of a ranch house. In each of these examples, the interface (function) of the object is specified, but the implementation details (e.g. what are the sizes of the bedrooms?) are not specified completely.

Versions can have two distinct forms of attributes, those inherited from the object type, and those defined to be unique for each version. Attributes inherited from the object type reproduce the

interface characteristics of the object type. Attributes defined to be version specific are the attributes which distinguish one version of a particular type from another version of the same type. Another way of describing version generalization is that it is a form of abstraction in which similar objects are related to a higher level object.

Instantiation [BAT085] occurs when an object is copied. Creating multiple instances of an object provides for distinction between the various copies. Both object types and object versions can be instantiated. The purpose of instantiating will be extended to provide meanings for instances of type and version. A version will be instantiated to provide a local working copy of a previous design, which may be plugged to any level of detail. Types (or subtypes) will be instantiated to produce a working copy for design work from scratch, in cases where no existing design can be used. Figure 2 shows an object O1 which is an instance of type A. O1 would be produced to provide a working copy of type A as a starting point in this particular design. The fact that O1 is instantiated from its parent type tells us that the implementation specifications for the final product are not available and will be developed from scratch. If O1 were instantiated from V1 instead, the design would begin from the point in V1 where implementation details left off, indicating that some similarity exists between the implementation of O1 and V1.

A hierarchy is formed for the set of designs for a particular type/subtype, and is called a version hierarchy. In this hierarchy, going from a higher level to the next lower level, we find that more implementation details are specified. The difference between the type/subtype generalization and the version hierarchy is that different versions of an object have the same set of attributes, and not necessarily the same values, while different types (or subtypes) will have different sets of attributes from each other. Figure 3 depicts two version hierarchies. In this case, ranch and colonial are subtypes of type house. Each subtype can have its own version hierarchy. The blocks labelled two bedroom, three bedroom, and four bedroom are on the same level in the diagram because they represent

mutually exclusive versions. Each block in the diagram is a potential starting point for future designs.

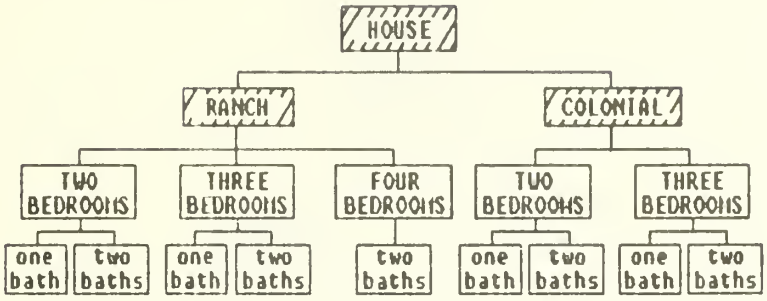



figure 3

An extension of the instantiation abstraction is the instance hierarchy. The purpose of this hierarchy is to record different design alternatives which are produced in the design process. Figure 4 is an example of an instance hierarchy for a house being designed for John Jones. Since Mr Jones is building this house from scratch, the design starting point was an instantiation from subtype ranch. In the course of designing his house, Mr Jones wasn't sure whether he wanted an attached or detached garage, two alternatives represented in the hierarchy. The reason for saving the hierarchy is that Mr Jones may decide on an attached garage, finish the design, and then change his mind. The hierarchy would permit him to go back to the point of the detached garage and re-complete the design. All of the information provided in the original design would be usable in the second design except for information about the garage itself.

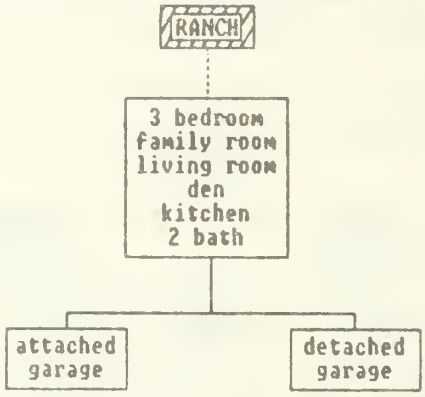



figure 4

Figure 5 provides a conceptual schema of the house design and construction example. The model shown represents the hierarchy of type aggregations for a generic house.

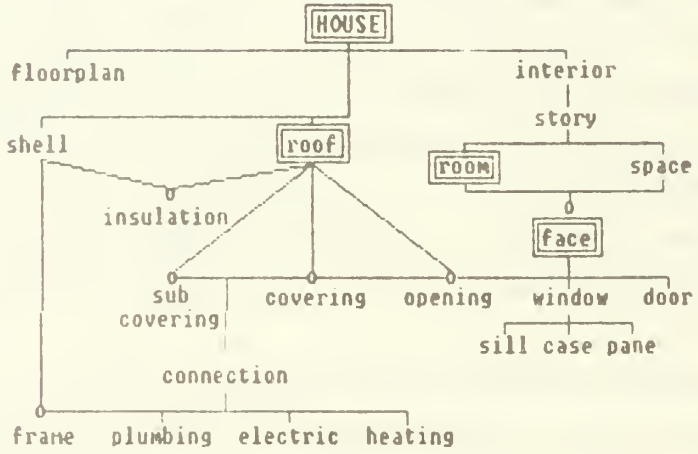



figure 5

House is the aggregation of a floor plan, a shell, a roof, and an interior. Each of shell, roof, and interior are further defined as aggregations of objects, some of which are shared. For example, both roof and shell have a component called "insulation".

The bubble notation in figure 5 represents an exclusive-or relationship among the types involved. A particular instance of insulation is either of type shell or roof, but not both. The

insulation associated with the shell would be a separate instance and version from the insulation associated with the roof. The double rectangle notation represents types which have named subtypes. For example, room has subtypes named kitchen, den, bathroom, bedroom, etc., which can be instantiated to produce a specific configuration.

Figure 6 summarizes the relationships between type, subtypes, versions, version hierarchies, and instances.

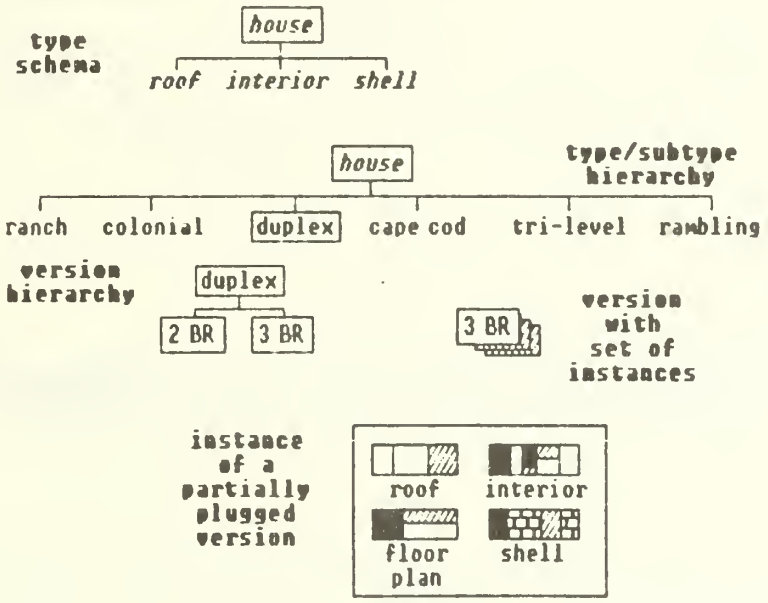



figure 6

## User Interface

For an integrated Information System to be really useful to the users of the system, it must have a user interface which enhances productivity. To the best of our knowledge, no other publications have been found which fully address this issue. We believe we are the first to address the issue of a user interface which appeals to such a wide range of users. Several interfaces exist, but most database interfaces are oriented towards the more advanced users. The feature which we feel is most desirable in a user interface is the use of graphics to appeal to users of all ranges of expertise. Since objects manipulated in this application are complex, we feel an extension of a conventional textual, statement-by-statement language would be

difficult to use and highly error-prone. A graphical, icon-based menu interface is more appropriate and natural for this application. Using a graphical approach, a diagram is used to show the structure of a database schema, and users directly manipulate the diagram to update and query the database. The actual content of the database can be displayed either textually or graphically.

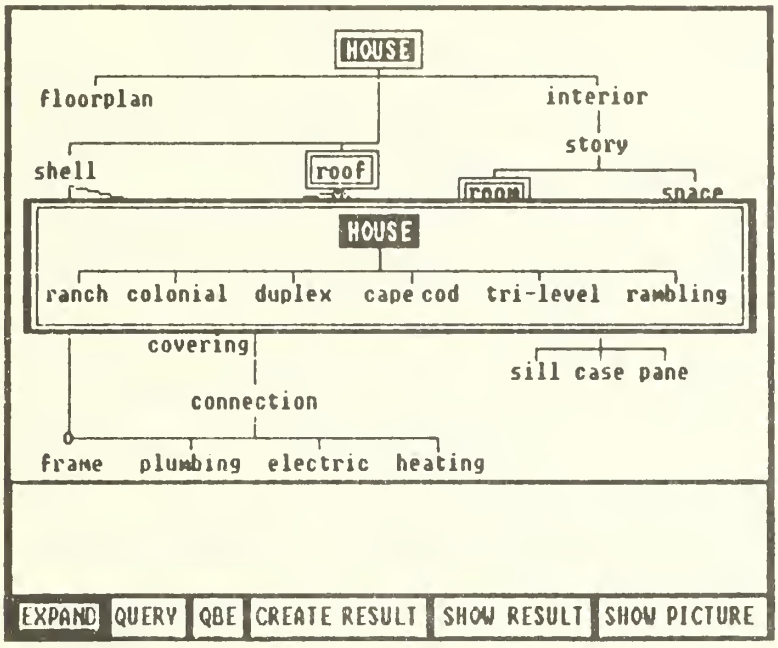
The Graphics LAnguage for Database (GLAD) [WU86] is proposed as the graphics interface for manipulating advanced application databases. Two outstanding advantages of GLAD are: first, it provides a single environment where different techniques proposed for different purposes can be integrated. Second, GLAD maintains a high degree of data independence, and as such, can be used as a front-end to a relational or network DBMS.

We have adopted the GLAD interface with some modifications and extensions. The EXPAND, DESCRIBE, and QUERY operations provided by GLAD will form the base for operations in our data model. The EXPAND operation provides a pop-up menu which displays alternatives for expanding generalized objects. DESCRIBE provides users with detailed explanations about objects, relationships between objects, and results of queries. QUERY has a QBE-like interface for specifying a query, which provides the user with the capability for expressing relational algebra or relational calculus equivalent queries.

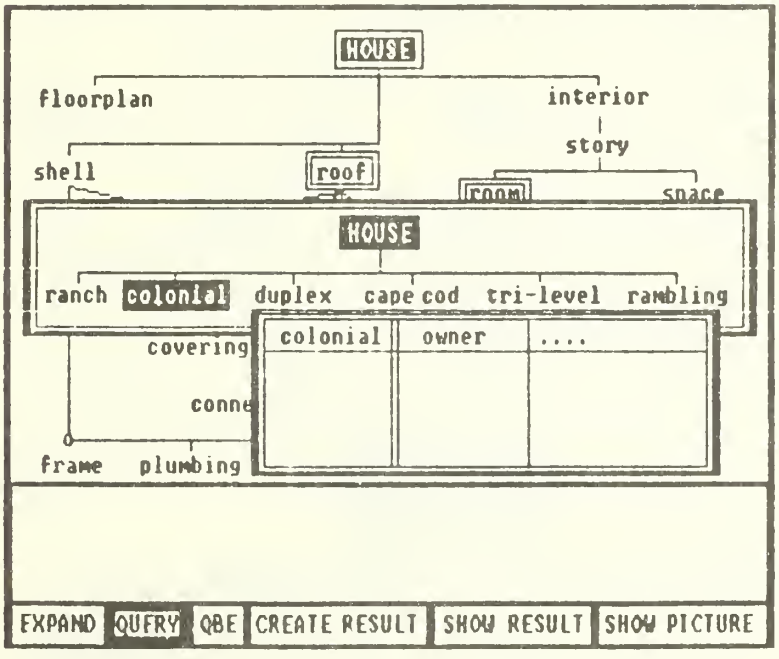
We will demonstrate how a modified GLAD interface can be used by going through a sample session. Suppose a user wants to know all owners of colonial houses. The following sequence will demonstrate the query operation with several operations:

Query 1) Who are the owners of colonial-type houses?

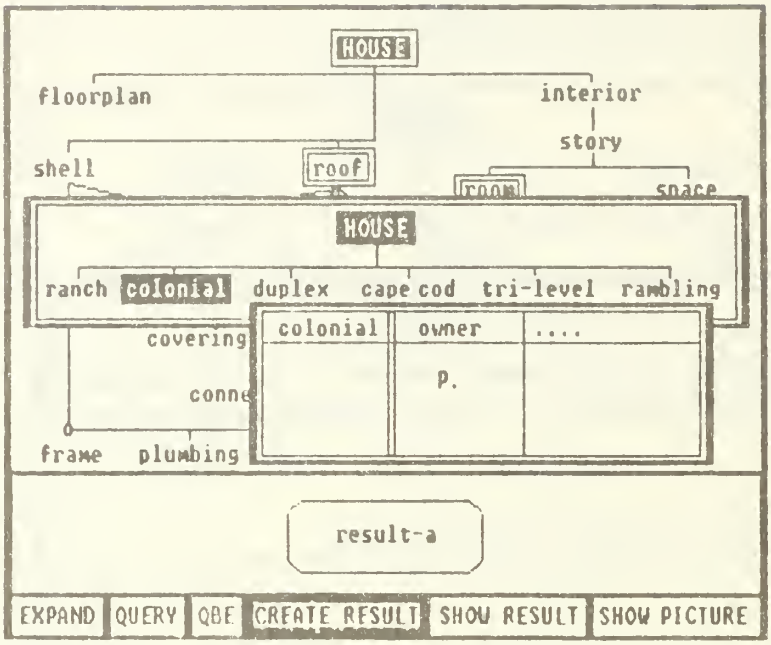
select house and expand



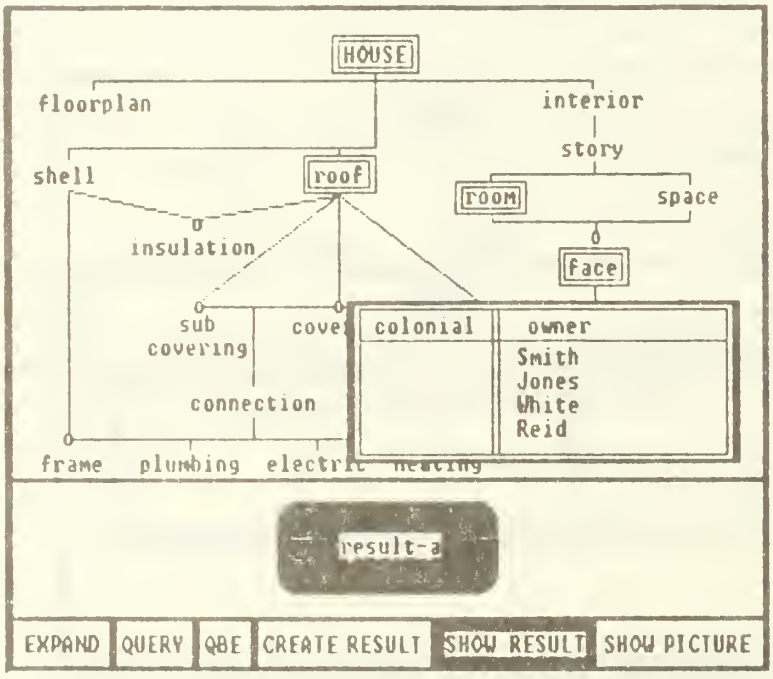

select colonial and query



type in specification, select create result

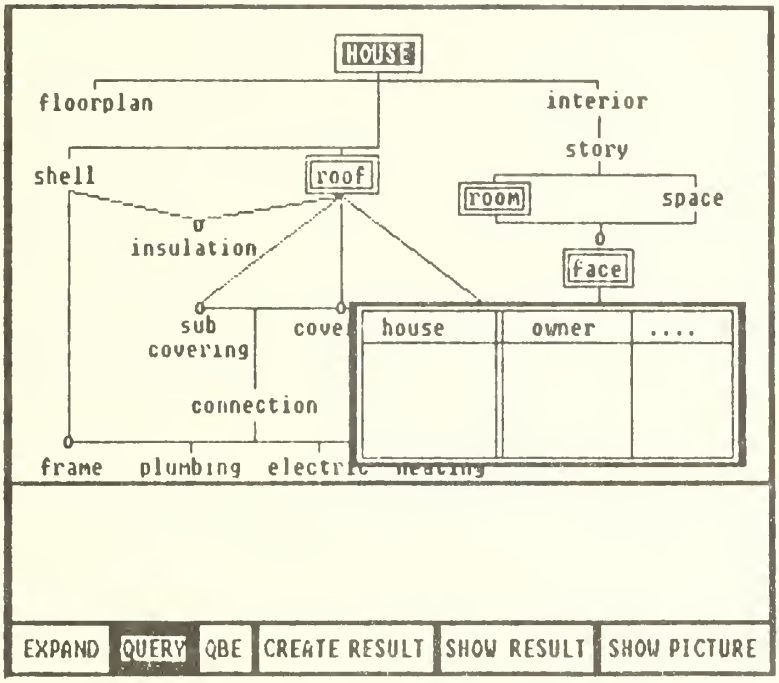



select result-a, use show result to see the content

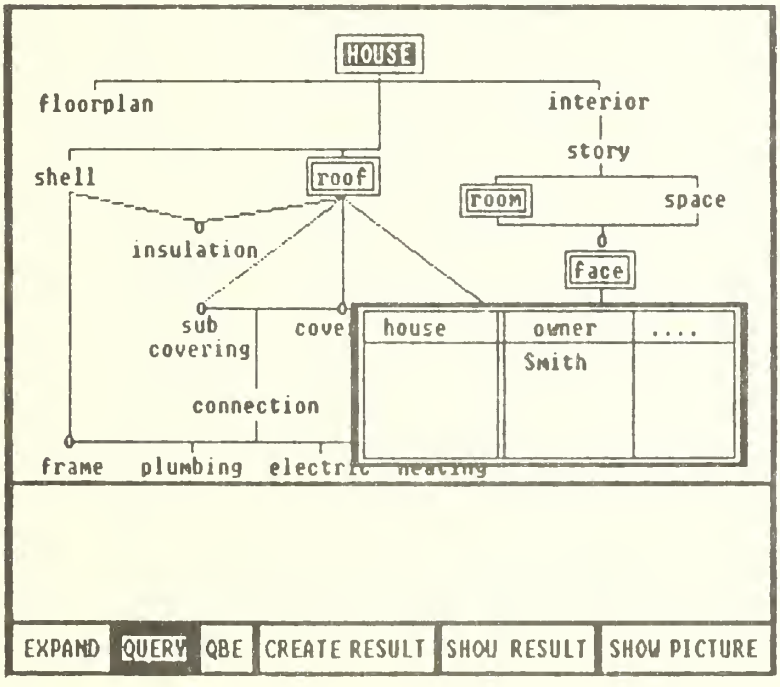


Query 2) Get 3-D view of roof of Smith's house

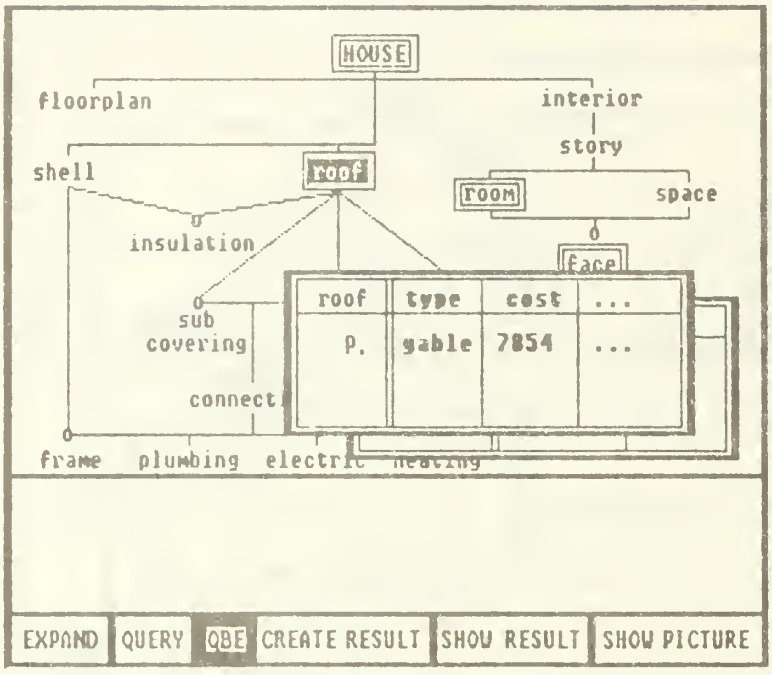
select house and query




type in Smith

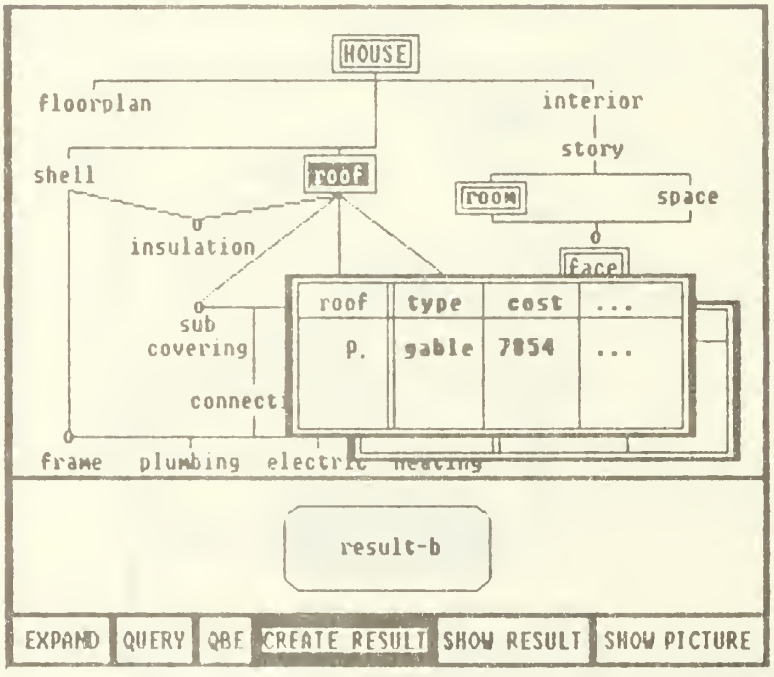


select roof and qbe and type p.

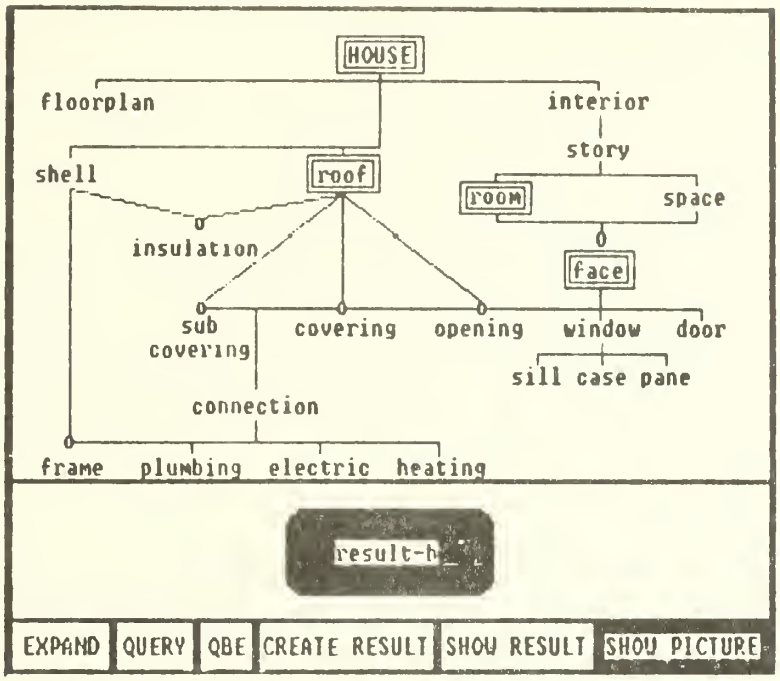



This will give us all the information about the roof of Smith's house.

select create result



select result-b and show picture




Extensions to the GLAD operations will include the SHOW VERSIONS, SHOW INSTANCES, and SHOW PICTURE operations. The SHOW VERSIONS operation will display the version hierarchy for a given type or subtype defined in the system. SHOW INSTANCES will permit users to view information on designs currently in progress as well as those which have been completed. The SHOW PICTURE operation as illustrated above provides a graphical display of an object. This will be accomplished by storing the world coordinates for each instance of an object as an attribute of that object. A prototypical picture is produced for a version of a type or subtype. Standard three-dimensional operations such as zoom, rotate, etc., will be available to enhance the graphical display feature. The initial implementation of our model will include a static viewing capability with the SHOW PICTURE operation. In a future implementation, we will extend this feature to include a dynamic viewing, which will permit a three-dimensional walkthrough of a design. Another feature which is envisioned is the ability to go from a high-level three-dimensional view of an object, using the zoom

function, to a lower-level object, where the DESCRIBE operation will display the textual attribute information about the lower-level object.

## Conclusion

Two major requirements of a fully integrated information support system for advanced application environments are an ability to capture the semantics of the environment and a good user interface. The semantics can be represented in such a system using appropriate abstraction concepts. The use of graphics in a user interface is a must.

A set of abstraction concepts and graphical user interface have been proposed. The abstraction concepts are extensions of existing concepts. The user interface is an extended GLAD interface. These abstraction concepts and user interface are described by use of a house construction example, with a goal of developing a completely integrated information support system for more generic design and manufacturing processes. In such a system, the product life cycle is divided into two phases, which we will call design and manufacturing. The design phase includes formulation of an image of the final product without the need for implementation details. Once the notion of physical appearance has been developed, the design is completed by supplying implementation details. By-products of the design phase could include an estimated cost and bill-of-materials for the final product. The manufacturing phase would include production and assembly of the system components. This paper describes the model for the design phase. A paper on the manufacturing phase is included in future research initiatives.

Another possible area for future research deals with possible internal representations of design objects using syntax directed editors with context-free [RAB185] or context-sensitive [LEE83] grammar rules. The use of such an editor will ensure that the design satisfies integrity constraints (which will keep a window from being

placed in a floor). The rules of the grammar would be applied step by step to obtain a specific design.

We intend to pursue the object-oriented approach to data management. This approach has been successfully applied to a wide range of applications, including CAD systems, office information systems, and knowledge representation systems. In these environments, the concept has been shown to be extremely versatile and flexible. It is our belief that it can be extended to our generic design and manufacturing domain, and will be well-suited to that type of constantly changing environment.

The graphics capability required for our system will be provided by use of either a Silicon Graphics Iris or an NBI ISI workstation. The Iris is a M68010-based workstation with 3Mb of cpu memory with a 1024 x 788 x 8 bit display memory. The ISI is a VME 68020-based machine with 2Mb of cpu memory with a VGSM graphics subsystem.

## References


[BATO85] BATORY, D.S. and KIM, W. Modeling Concepts for VLSI CAD Objects. *ACM Trans. Database Syst. 10,3* (Sep. 1985), 322-346.

[BROD82] BRODY, M.L. On the Development of Data Models. *On Conceptual Modeling*, Springer-Verlag, 1982, 19-47.

[CHEN76] CHEN, P.P.S. The Entity-Relationship Model-Toward a Unified View of Data. *ACM Trans. Database Syst. 1,1* (Mar. 1976),9-36.

[HARD86] HARDWICK, M. and SINHA, G. A Data Management System for Graphical Objects. *IEEE International Conference on Data Engineering*, 1986, 447-455.

[LEE83] LEE, Y.C. and FU, K.S. A CSG Based DBMS for CAD/CAM and its Supporting Query Language. *Databases for Engineering Applications*, 1983, 123-130.

[MCLE83] MCLEOD, D., NARAYANASWAMY, K., BAPA RAO, K. V. An Approach to Information Management for CAD/VLSI Applications. *Databases for Engineering Applications*, 1983, 39-50.

[RABI85] RABITTI, F. A Model for Multimedia Documents. *Office Automation*, Springer-Verlag, 1985, 227-250.

[SMIT77] SMITH, J.M. and SMITH, D.C.P Database Abstractions: Aggregation and Generalization. *ACM Trans. Database Syst. 2,2* (Jun 1977), 105-133

[WU86] WU, C.T. A Unified Interface Method for Interacting with a Database. *Submitted for publication.*